

PATTERN MATCHING (Tasarım eşleştirmeleri)

Akış kontrol mekanizmalarında yapabildiğimiz şeylerin daha da desenleşmiş halidir. Olayı daha da kolaylaştırmamızı sağlar.

- C# 7.0 ile
 - ▼ Type matching
 - Object içerisindeki bir tipin belirlenmesinde kullanılan "is" operatörünün desenleştirilmiş halidir.
 - is ile belirlenen türün direkt dönüşümünü sağlar

```

object x = 125;
if (x is string)
{
    string xx = x as string;
    Console.WriteLine($"x değişkeni string tipindedir.");
}
else if (x is int)
{
    int xx = (int)x;
    Console.WriteLine($"x değişkeni int tipindedir.");
}

```

```

object x = 125;

if (x is string xx)
    Console.WriteLine($"x değişkeni string tipindedir.");
else if (x is int xx)
    Console.WriteLine($"x değişkeni int tipindedir.");

```

-

```

object x= "irem";
if (x is string a)
{

}

```

Console.WriteLine(a); // a ya buradan erişemeyiz.

// Çünkü; type pattern da tanımlanan değişkenlere manuel olarak erişilebilir.

```
// null olan değişkenler herhangi bir yerden çağırılamaz.  
//a nın null olma ihtimalinin sebebi ise, x ya string değ
```

-

▼ Constant Pattern

- Elimizdeki veriyi sabit bir değer ile karşılaştırabilmemizi sağlar. == kontrolünü sağlar kısaca.

-

```
object x=123; //kontrol edeceğimiz değer object olmak zorundadır.  
if(x is 123) //değer bazlı bir kontroldür. Eğer değer kontrol edilir  
{  
}
```

```
if (x is int) //tür bazlı bir kontroldür. Eğer tür kontrol edilir  
{  
}
```

-

▼ var Pattern

- Eldeki veriyi var değişkeni ile elde etmemizi sağlamaktadır.
- **ÇOK ÖNEMLİ**: Normalde kullanılan var derleyici sürecinde türü belirlerken, burada kullanılan var türü runtime da belirler.

```
object x ="asdfg";  
if(x is var a) // x in türü her ne ise a ya unboxing enedilir  
{  
}
```

▼ Recursive Pattern

- Switch bloğunda referans türlü değişkenlerde kontrol edilmektedir.

- Case null komutu ile ilgili türün/ referansın null olup olmasını kontrol edebilmesinden dolayı Constant pattern'ı kapsamaktadır.

-

```
object x="lkdnclkn";
if(x is string a)
{
    Console.Write (a);
}

if (x is var b)
{
}

bool result = x is string o1;
Console.WriteLine (o1); // Hata verir. Type pattern 'da x


bool result2 = x is string o2;
Console.WriteLine (o2); // var pattern'da ise değişkenin (
```

-

- C# 9.0 ile

- ▼ Simple type pattern

- Değişken isimleri zorunluluğunu kaldırır.



```
object obj = new Person();
switch (obj)
{
    case Person p:
        //...
        break;
}
```

```
object obj = new Person();
switch (obj)
{
    case Person:
        //...
        break;
}
```

Simple Type Patterns

- Bir değişken içerisindeki değerin belirli bir türde olup olmadığını hızlı bir şekilde kontrol etmemizi sağlayan desendir.
- C# 9.0'dan önce Type Pattern ile yapılan tür bildirimlerinin yanına değişken adı tanımlaması yahut discard ifadesinin kullanılması zaruriydi. C# 9.0 ile bu gereksiz zorunluluk ortadan kaldırılmış ve direkt olarak tür kontrol işlemine odaklanılması sağlanmıştır.

▼ Relations Pattern



Relational Patterns

- Desenlerde <, >, <= ve >= operatörleri kullanılabilmekte ve belirli karşılaştırmalar hızlıca gerçekleştirilebilmektedir.

```
int number = 111;
string result = number switch
{
    < 50 => "50'den küçük",
    > 50 => "50'den büyük",
    _ => "Hiçbiri"
};
```

```
int number = 111;
string result = number switch
{
    < 50 => "50'den küçük",
    > 50 => "50'den büyük",
    50 => "50'ye eşit"
};
```

Bir mülakat sorusu olabilir. ⇒ Switch case ile if arasındaki fark nedir?

Switch case sadece eşitliğe bakarken C# 9.0 yapısı ile gelen Relational patterns ile diğer karşılaştırmaları da yapabilmekteyiz. if yapısı tüm karşılaştırma ve eşitliklerde kullanılır

▼ Logical Pattern

And or ve not gibi mantıksal ifadeler kullanılabilmektedir.

```
int number = 60;
string result = number switch
{
    > 10 and < 50 => "10'dan büyük 50'den küçük",
    > 50 or < 100 and 60 => "50'den büyük veya 100'den küçük ve 60'a eşit",
    not 51 => "51 değil"
};
```

Relational Pattern ile oldukça uyumludur.

▼ Not pattern

Not operatörü ile kullanılır.

```
string GetProduct(IProduct p) => p switch
{
    Technologic => "Teknolojik",
    Computer => "Bilgisayar",
    not Goggles => "Gözlük"
};
```

```
object obj = new Goggles();
if (obj is not Technologic)
{
    //...
}
```

Not Patterns

not operatörünün kullanılabildiği bir desendir.